Beyond HTML framesets: using CSS to mimic the navigational features of HTML frames

by Mark Carlson

What a difference a few years makes. Even though the original Cascading Style Sheets (CSS) specification has been with us for almost ten years now, it has only been within the last several years that comprehensive support for it has been implemented by all of the major WWW browsers. HTML frameset documents (frames) provide many nice navigational features but these handy features come at a sometimes high price. Although some of the original problems (creating bookmarks, breaking out of frames, printing, following links, etc.) with HTML frames have been addressed by the major browsers, they are still cumbersome and problematic.

In this article, I will show you how to mimic a HTML frameset document using only standard HTML and CSS. This will allow you to take advantage of the navigational features that a HTML frameset document provides without some of the typical drawbacks.

The following link shows an example of a framed page utilizing only standard HTML tags and CSS:

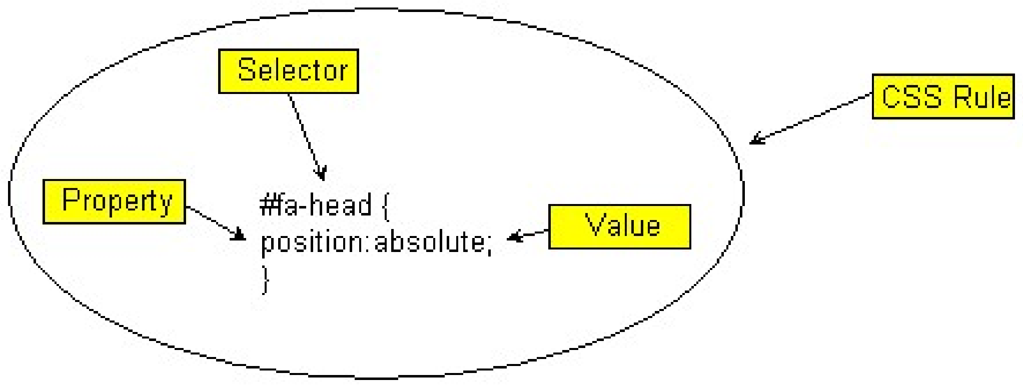
http://www.lib.washington.edu/specialcoll/findaids/docs/papersrecords/McGilvraJohn4806.xml

Click on the [+] next to the Inventory link in the table of contents bar on the left side of the page to expand the listing. Notice that the table of contents scrolls separately from the main "body" section. The links from the table of contents to the "body" section are standard HTML anchor tags. The "target" attributes specifying where links should display, which are usually required when using frames, are not necessary and the whole document is built as a single HTML document.

So how does it all work?

Before we get started let me first say that this document assumes that the reader has some knowledge of how to use CSS within their HTML documents. If you need assistance in this area, check out the following introduction to CSS: http://www.w3.org/TR/CSS21/intro.html

Here is a quick overview of the parts of a typical CSS "rule" so that you'll understand the CSS terminology that I use throughout this document.



```
#fa-head {
position:absolute;
}
```

The CSS property that makes independently scrolling sections of the page possible is "overflow". This first appeared in the CSS2 specification in 1998, but didn't have widespread support until fairly recently. Setting the "overflow" property to "auto" or "scroll" allows the various sections of the page to scroll independently.

In the example document references above, I have four sections or "frames" that the document is divided into, although I'll only be addressing three in this article. [The fourth frame is for the "context" view at the bottom of the page.] The top frame encompasses the header section of the page, the table of contents makes up the second frame and the body or content section makes up the final frame. Using CSS, we set these up for our document like this:

```
#fa-head {
position:absolute;
margin:0px;
top:0px;
left:0px;
display:block;
width:100%;
height:120px;
color:#000;
z-index: 2;
}

#fa-toc {
position:absolute;
height: auto;
left:0px;
top:120px;
bottom:0px;
width: 180px;
overflow:auto;
background:#7789A1;
z-index: 1;
}

#fa-content {
position:absolute;
left:180px;
top:120px;
bottom:0px;
overflow:auto;
z-index: 1;
}
```

In CSS jargon, the ID selectors #fa-head, #fa-toc and #fa-content correspond to the layout of the head, table of contents and body sections of the document respectively. Notice that within the #fa-toc and #fa-content sections, the overflow property is set to "auto". This sets scrollbars to appear when necessary, when the content of each section or "frame" would display beyond the borders of the browser window. If you want scrollbars to appear at all times, set the value to "scroll". These are the "generic" settings for the layout of the page that apply for all browsers. While CSS support has come a long way, the lack of full implementation and varying interpretations of the specification still means that you need to make "per browser" adjustments in some cases. So beyond this, we just need to handle the various browsers' differing implementations of CSS to get everything to display properly.

For the CSS frames layout to work in Internet Explorer (IE), you must set the "overflow" property to "hidden" for both the html and body selectors. These settings aren't required for any of the other major browsers, but the other browsers just ignore these IE specific declarations so we can just include those in the general CSS layout for the page.

```
html {
height:100%;
max-height:100%;
padding:0px;
margin:0px;
border:0px;
background:#fff;
overflow: hidden;
}

body {
height: 100%;
max-height:100%;
overflow:hidden;
padding:0px;
margin:0px;
border:0px;
}
```

In addition, you must also set the following CSS rules for correct rendering in IE:

```
* html #fa-toc {
height:100%;
top:0px;
bottom:0px;
border-top:120px solid #fff;
}

* html #fa-content {
height:100%;
top:0px;
bottom:0px;
border-top:120px solid #fff;
}
```

The values for the declarations within these rules should correspond to the values for the #fa-toc and #fa-content ID selectors above. For instance, if you have:

```
#fa-toc {
top: 90px;
}
```

You should have:

```
* html #fa-toc {
border-top: 90px solid #fff;
}
```

The same thing holds true for the #fa-content selector, where the "top" property value should be set to the same value as that of the "border-top" property value.

The only other rule that is pertinent to the layout of the page is this class selector:

```
.fa-inner {
display:block;
padding:10px 10px 10px 10px;
}
```

This is used within the various ID selectors to give us a padding of 10 pixels around the borders of the various frames.

Let's do a quick mock up of a HTML page incorporating the information covered so far:

```
<html>
<head>
<title>Quick HTML markup from mock frames</title>
<style type="text/css" media="screen">
html {
height:100%;
max-height:100%;
padding:0px;
margin:0px;
border:0px;
background:#fff;
overflow: hidden;
}

body {
height: 100%;
max-height:100%;
overflow:hidden;
padding:0px;
margin:0px;
border:0px;
}

#fa-head {
position:absolute;
margin:0px;
top:0px;
left:0px;
display:block;
width:100%;
height:120px;
color:#000;
z-index: 2;
}
```

```
#fa-toc {
position:absolute;
height: auto;
left:0px;
top:120px;
bottom:0px;
width: 180px;
overflow:auto;
background:#7789A1;
z-index: 1;
}

#fa-content {
position:absolute;
left:180px;
top:120px;
bottom:105px;
overflow:auto;
cursor: hand;
z-index: 1;
}

* html #fa-toc {
height:100%;
top:0px;
bottom:0px;
border-top:120px solid #fff;
}

* html #fa-content {
height:100%;
top:0px;
bottom:0px;
border-top:120px solid #fff;
}

.fa-inner {
display:block;
padding:10px 10px 10px 10px;
}
</style>
</head>
<body>
<div id="fa-head">
This is our header section
</div>
<div id="fa-toc">
<div class="fa-inner">
This is the table of contents
</div>
</div>
<div id="fa-content">
```

```
<div class="fa-inner">
This is the body of our document
</div>
</div>
</body>
</html>
```

The result of this looks like this

So now you are basically 90% there. The only other things to do are to replace the mock up text between each of the <div> or combination of <div> elements with your content and account for variations for other browsers.

As I mentioned earlier, the various browsers implement the CSS specification differently in some cases, so you need to adjust for this. I use a simple JavaScript that detects the particular browser that the client is using and sets the CSS styles that are appropriate for that browser. I made one of two assumptions when formulating this script: 1) people would be using the browser version available at the time of creating these browser specific settings, or, 2) even if they weren't using the latest version, the settings that work with the latest version are likely to work with earlier versions (to a certain point, of course) since the CSS implementations for browsers tend to be backwards compatible.   So this script was built based on these particular browser versions:

Firefox 1.5+
Internet Explorer 6+
Mozilla Suite 1.7+
Netscape 7.2+
Opera8.5+

Detecting the client's browser has become more convoluted over the years. The script that I am including here is not without its particular pitfalls and tradeoffs (for instance, there is currently no support for Mac's Safari browser) but it does work with the above list of browsers. Certainly not a perfect solution in all cases, but I consider it a "work in progress" and am always looking for ways to improve and enhance things. So here's the code:

```
<script language="javascript" type="text/javascript">
var myBrowser = navigator.userAgent;
if (myBrowser.indexOf("Opera") != -1)  {
document.write('<style type="text/css" media="screen">#fa-content {right: 0; width: auto;} #fa-toc {position: absolute; height:auto; }</style>')
} else {
if (myBrowser.indexOf("Firefox") != -1) {
document.write('<style type="text/css" media="screen">#fa-content {height: auto; width: auto; right: 0; position: fixed;}</style>')
} else {
if (myBrowser.indexOf("Netscape") != -1) {
document.write('<style type="text/css" media="screen">#fa-content {width: auto; right: 0; position: fixed;} #fa-toc {height:auto;}</style>')
} else {
//MSIE code must be placed after Opera code
if (myBrowser.indexOf("MSIE") != -1) {
//i.e. Internet Explorer
document.write('<style type="text/css" media="screen">.fa-inner {width: 100%;} #fa-content {width:expression(document.body.clientWidth - 180)}</style>')
} else {
```

```
//Mozilla navigator must be last to avoid false hits on other browsers
if (myBrowser.indexOf("Mozilla") != -1) {
document.write('<style type="text/css" media="screen">#fa-content {width: auto; right: 0; position: fixed;} #fa-toc {height:auto;}</style>')
} else {
document.write('<style type="text/css" media="screen">#fa-content {width: auto; right: 0; position: fixed;} #fa-toc {height:auto;}</style>')
alert('Note: Your browser may not support the display features of this page. For best results, use one of the following browsers:\n\nFirefox 1.5+\nInternet Explorer 6+\nMozilla Suite 1.7+\nNetscape 7.2+\nOpera8.5+\n')
}
}
}
}
}
</script>
```

Many of the other browsers available out there utilize the rendering engine of one of the major browsers, so the correct settings will be added for the appropriate rendering engine that they have on their computer. You can always add additional support for other browsers as needed.

You might note that the browser specific settings are the same for some of the browsers. For instance, the settings for "Netscape" are the same as "Mozilla". The reason for this is mainly a troubleshooting one. It was easiest to tackle the browser specific settings one browser at a time, so the script reflects this process.

In the browser-specific settings for IE (MSIE) above, note that one of the other IE-specific fixes that is required in this layout is that you need to deduct the number of pixels from the "#fa-content" section that is equal to the number of pixels of the left hand margin specified in the #fa-content rule. You then need to include this value in a CSS property that only IE supports. For instance, the left margin of the #fa-content rule is set to 180 pixels:

```
left:180px
```

So we use that value and subtract it from the expression that is triggered by the above script when IE is encountered:

```
#fa-content {
width:expression(document.body.clientWidth - 180)
}
```

The one drawback to using scripted browser specific settings is that it requires that active scripting be enabled in the client’s browser. Unless you are an advanced computer user and have deliberately turned it off, active scripting is going to be enabled by default. So what happens if it is turned off for some reason? Then you’ll need a "generic" script that will render an acceptable display in any browser when active scripting is turned off. You place that between <noscript> tags in HTML. Since the variable width layout is the cause of all of the problems for the various browsers, you can set the width of the "#fa-content" frame to a fixed width:

```
<noscript>
<style type="text/css" media="screen">
#fa-content {width: 620px;}
</style>
</noscript>
```

Note that I have included the "media" attribute for the <style> element in HTML. Because of the unusual IE-specific settings, we also need to make adjustments on print output. So we also need a separate "print" CSS stylesheet. In the following example, I have omitted the printing of the header and table of contents on print output by setting the "display" property to the value "none". The .fa-inner class selector sets a static width of 650 pixels:

```
<style type="text/css" media="print">
body {margin: 0px; width: auto;}
html {margin: 0px; width: auto;}

* {font-family: verdana, sans-serif; font-size: 10pt;}
#fa-toc {display: none;}
#fa-head {display: none;}
.fa-inner {width: 650px;}
#fa-content {width: auto;}
</style>
```

This print stylesheet renders an acceptable print output for all of the major browsers.

One last thing to mention that I noticed is that a couple of the browsers are lazy when it comes to deciding when to display a scrollbar and sometimes the scrollbar won't scroll far enough to show all of the text for a particular section or "frame". If you encounter this problem, the fix for this is to add a few lines of "padding" by inserting a series of <br/> tags at the end of the content for that section.

The final step is to make sure that all of the code is loaded in the proper order when the document is loading. To eliminate potential problems, place stylesheets and scripts in the following order in your HTML document:

1. "Screen" CSS
2. "Print" CSS
3. Browser specific rendering javascript
4. <noscript> CSS

These pages may give you additional ideas:

http://www.cssplay.co.uk/
http://tutorials.alsacreations.com/frames/frames2.php?page=accueil
http://www.webreference.com/programming/css_frames/

If you have questions or feedback about this page, feel free to contact me

Last modified: November 1, 2006